

University of Maine

From the Selected Works of Radek Glaser

Spring May 10, 2007

Design and Characteristics of a Split Hopkinson Pressure Bar Apparatus

Radek Glaser, *University of Maine - Main*

Jesse Haines, *University of Maine*

Christopher Knight, *University of Maine*



Available at: https://works.bepress.com/radek_glaser/2/

DESIGN AND CHARACTERISTICS OF A SPLIT HOPKINSON PRESSURE BAR APPARATUS

**Jesse Haines
Christopher Knight
Radek Glaser**

May 10th, 2007

**University Of Maine
Mechanical Engineering
5711 Boardman Hall
Orono, ME 04469**

ABSTRACT

A Split Hopkinson Bar Apparatus, also known as Kolsky Bar that is capable of conducting compressive strain rate testing in the approximate ranges from 50 to 10^4 in. /in. /sec. was designed as a part of a Senior Design Project. Generally, this device is similar to that first used by Kolsky in 1949. The design of this device is presented here in two stages:

1. Research, design and manufacturing of the Stress Generating System
2. Experimental Part – Testing of the apparatus to obtain necessary data.

The present phase of the design was focused mostly on the stress generating system for the apparatus. Final experiments were conducted to obtain the calibration curve and the shapes of generated waves. Not only the description of the design process is presented, but also the final details including the assembly and testing are given. Also some valuable suggestions for future work that is necessary to the make the apparatus better is given.

ACKNOWLEDGMENTS

We would like to express our thanks for the assistance and help we received from Michael Knight, our advisor Prof. Vincent Caccese, Arthur Pete, Prof. Michael Peterson and the Staff of the Advanced Manufacturing Center during the lifetime of the Split Hopkinson Pressure Bar Project.

Table of Contents

<u>Title</u>	<u>Page</u>
Abstract.....	1
Acknowledgments.....	1
Introduction.....	3
Objectives of the Project.....	3
Researching the Split Hopkinson Bar Apparatus	3
Design Approach.....	4
Anticipated results.....	4
Final Design.....	5
Design Overview.....	5
Split Hopkinson Bar Parts.....	7
Control System.....	9
Design Testing-Experimental.....	12
Results and Conclusion.....	15
Recommendations.....	15
Appendix 1-Abandoned ideas.....	16
Appendix 2-Additional graphs of the waves.....	18
Appendix 3- Analog and Digital Pin IN/OUT connectors.....	20
Appendix 4-Program Codes.....	22
Endnotes.....	22

INTRODUCTION

The Split Hopkinson Pressure Bar design began during the Fall semester of 2006 in the Mechanical Engineering Senior Design class at the University Of Maine. Our group that consisted of Chris Knight, Jesse Haines, and Radek Glaser chose the project based not only on its high potential and long time use, but also because of its multiple capabilities. The Split Hopkinson Bar has the ability to test the compressive strength of variety materials, such as: laminated composites, ceramics, different metals and ballistics. The apparatus is being designed with longevity and flexibility of the operational characteristics in mind. This will allow for a wide range of testing and limit future costs associated with the project. Evidently, Split Hopkinson Pressure Bar Apparatus Project has the potential for future growth through an external or departmental funding associated with the testing.

OBJECTIVES OF THE PROJECT

During the project the main focus was placed on the design, assembly and testing of the stress generating system of the Split Hopkinson Pressure Bar Apparatus. The

following were the main objectives of the design process of the apparatus:

Design and build a fully functional apparatus that would be capable of:

- * Operating in a safe manner
- * Producing striker velocities in the range of 100 -160 ft/sec.
- * Producing pressure-velocity calibration curve
- * Generating impact-compression tests at strain rates ranging from 50 to 10^4 in./in./sec.
- * Generating propagation waves that can be used to determine strain-stress relationship

RESEARCHING THE SPLIT HOPKINSON BAR APPARATUS

Early in the design class during the initial stages of our Split Hopkinson Bar Apparatus Design we attempted to find as much useful information as possible about the apparatus. Our main task was to find the most reliable information that could lead us to successful completion of our project including all our objectives. The main concern was to find an efficient way to allow for a sudden flow of the compressed gas from the tank to the

launching tube so the striker could be accelerated properly. After researching not only many websites and publications on different versions of the Split Hopkinson Bars around the world, we discovered that in many cases this crucial information was not really available for public access. Finally, we found an unclassified army materials and mechanic research booklet titled “Design and Operating Characteristics of a Split Hopkinson Pressure Bar Apparatus”.¹ This booklet is where much of our own research and design information of our Split Hopkinson Pressure Bar Apparatus came from. From the research Phase we moved to design approach part that is included in the next section.

DESIGN APPROACH

As stated before, most of the details in the design of our Split Hopkinson Bar Apparatus came from the military booklet due to very meticulous information included in it. The booklet goes into large detail of the importance of the gun and its ability to perform very quick, large impulse, strain test. Although we tried to keep the details of the military design as best we could, soon we found out that we did not have the manufacturing capabilities or necessary budget. So we came with our own design that scaled down and simplified the original

apparatus. While designing the apparatus we kept the important parameters and characteristics of the design the same, to have equal or even better results. While the military design had 100 inch long launching tube we used 24 inch tube due to manufacturing issues. The original design used a solenoid valve that operated up to 300psi, but we increased operational pressure up to 750 psi by using appropriate solenoid valve² Also the frictional aspect was partially changed. Thanks to an explanation about potential problems with lubrication during the operation of the apparatus we were able to avoid the reduction of velocity that was due to the excess of lubrication. Being able to refer to the military design, not only made it much easier and faster for us to design our Split Hopkinson Pressure Bar Apparatus, but also helped us to avoid potentially unfavorable situations during the design process. Certain analytical procedures were conducted to get numerical results for the final design. These anticipated results follow in the next section.

ANTICIPATED RESULTS

In most of the designs we can expect that our results will fall into a certain range. In our case it came from computational model based on simple Newtonian dynamics

relationships with many details omitted to keep our model simple. The following are the steps that lead to final design.

Equations of Motion for Striker Bar

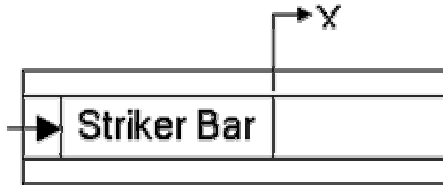


Fig1. Simple FBD

$$a_x = \frac{PA}{m} \quad x = \frac{1}{2} \frac{PA}{m} t^2 \quad v_x = \frac{PA}{m} t$$

$$t = \sqrt{\frac{2x}{PA/m}} \quad v_x = \sqrt{2 \frac{PA}{m} L}$$

Where the individual terms are:

v_x - velocity of the projectile,
P - initial pressure of compressed air tank,
A - area of the projectile,
L - travel distance (barrel length)
m - mass of the projectile
x - displacement of the projectile.
t - average travel time
a - acceleration

The values of **A** and **m** depend upon the radius and density ρ of the projectile itself.

Force Balance in the x direction, neglecting kinetic and air friction gives the following:

$$\sum F_x = PA = ma_{x-direction}$$

From the relationships above we obtained anticipated velocities and travel times for our

initial model. It is shown on the following graph.

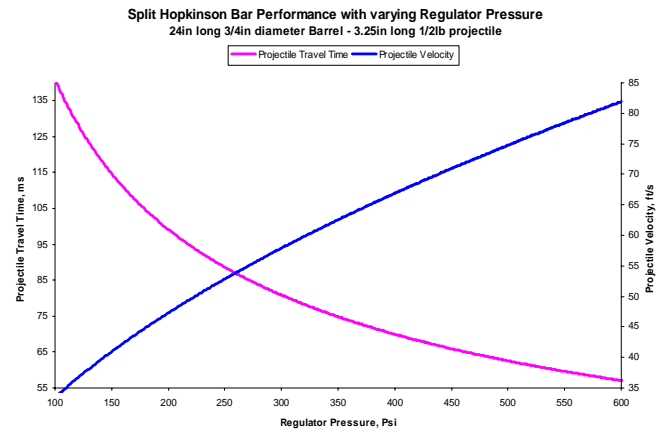


Fig2. Shows velocity profile (blue) and time of travel (pink).

These were anticipated results and the sections below will show how much the final results differ from the optimal.

FINAL DESIGN

DESIGN OVERVIEW

The Split Hopkinson Bar Apparatus works with compressed gas. The compressed nitrogen tank has an initial pressure of 2200 psig. There is a high pressure regulator that shows the current pressure in the tank and also limits the delivery pressure to the solenoid valve. The pressure of the exiting nitrogen can be set in variety of increments up to a maximum of 750 psi, which is the safe working pressure of the solenoid valve. Our computerized control system opens the

solenoid valve letting the gas expand to the launching tube pushing the striker out. When the striker reaches the end of the barrel it blocks the laser beam that is shining at the sensor, changing the voltage seen by the control system. At the same time striker bar impacts the incident bar producing the propagation wave through the bars. The control system closes the valve blocking off the delivery of nitrogen. The velocity of the striker is measured from the time the solenoid valve is opened to the time the voltage changes closing the solenoid valve. The excess of the pressure from the launching tube is released by a hand operated valve and then the vacuum suction is applied to move the striker bar to starting position.

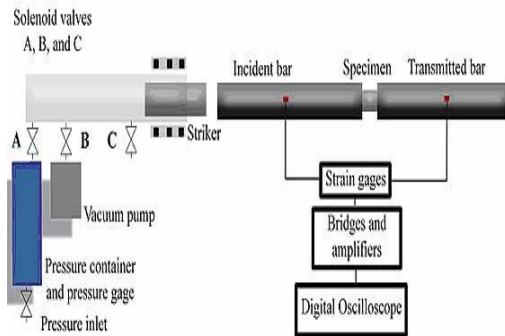


Fig.3 shows a diagram of the apparatus

The second part of the apparatus is the strain measuring section. It is composed of a striker bar, an incident bar, the test specimen and the output bar. The impact of the striker bar on incident bar produces a rectangular compression wave of well-defined amplitude and length in the incident bar.

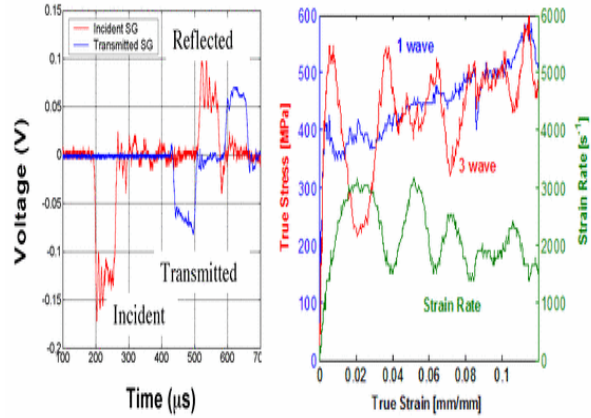


Fig4. Shows propagation waves on left and strains on right.

Then the wave travels through the incident bar, sample and to the output bar. Once this wave reaches the test specimen, part of it is transmitted through the specimen and part of it reflects back through the incident bar.

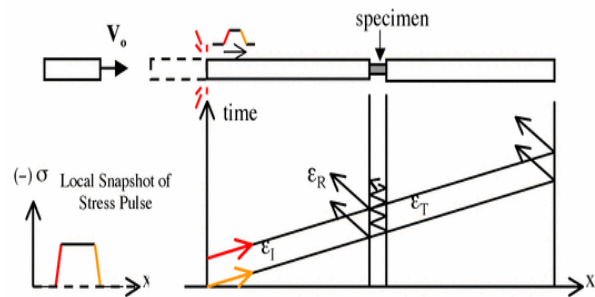


Fig5. Shows the refraction and absorption of the propagation waves.

Using one dimensional wave propagation analysis we can determine high strain rate stress-strain curves from the measurements of strain in the incident and output bars. In order to obtain correct wave propagation the striker bar must reach necessary velocity before it hits the incident bar. The wave is measured through a set of strain gages located on the incident and transmitter bar. The test specimen is located between the incident bar and

transmitter bar. Depending on the type of specimen and the impacting force we would get a part of the wave is transmitted back through the incident bar, and the rest is sent through the test specimen to the transmitter bar. Waves can be seen on the oscilloscope that is attached to the measuring system.

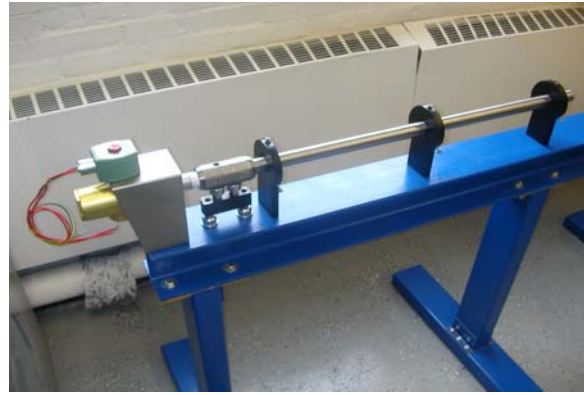


Fig5. Barrel, supports, flange and solenoid valve.

SPLIT HOPKINSON BAR PARTS

Launching Tube

A crucial component of the stress generating system is the launching tube. Table below shows two different barrels used for testing.

Gun Barrel Options			
TUBE	O. D. (in)	I. D. (in)	Length(in)
1	1.0 to .77	.61	33.0
2	1	0.731	24

Table 1.Shows Tube Options

The barrel is screwed onto a flange connecting it to the rest of the apparatus. Within the flange at the end of the barrel there is an inside taper that prevents the striker bar from over travel while being retracted by the vacuum pump. Therefore the geometry of the inside of the flange is changed resembling a converging nozzle. This definitely reduces the flow through the apparatus and pressure exerted on the striker bar. Figure below shows barrel, flange and solenoid valve currently being used during testing.

Striker Bars

Both striker bars were made of 4130 steel and have an outside diameter corresponding to the inside diameter of the launching tubes. The length of the striker bar that is currently being used is 3.125 inch, but this can vary depending on the desired propagation wavelength and the tube used for testing. Appropriate lubricant is be used sparingly inside the barrel and on the striker to reduce friction between these two parts during the discharge procedure.



Fig6. Striker bars.

Solenoid Valve, Gas Tank and Regulator

The most important component of the apparatus is the quick opening solenoid valve that releases compressed nitrogen which forces the striker down the barrel. The solenoid valve is an ASCO 8223G005³ that is rated for up to 750 psi. The compressed nitrogen comes from a nitrogen tank with an initial pressure of 2200 psi. The tank is connected to the solenoid valve through sealed piping and a pressure regulator. The pressure regulator allows us to deliver required flow by regulating the exiting delivery pressure up to a limiting pressure of 900psi. The solenoid valve releases the gas to launching tube and ultimately the striker exits with desired pressure and velocity. The maximum operating pressure for the valve is 750 psi.



Fig7. Gas tank, solenoid valve and pressure regulator.

Vacuum Pump

The vacuum pump is used to return the striker bar to the starting position at the rear

end of the barrel. The 0.4 W pump is connected to the flange that is connected to the barrel and solenoid. As soon as the solenoid valve shuts off the delivery of nitrogen to the launching tube and the excess gas is annually released from the tube, the vacuum pump is activated by the computer and creates vacuum in the barrel resulting in the bullet to be retracted to the starting position.



Fig8. Vacuum pump, tubing, and manual pressure valve.

Incident, Transmitter and Momentum Bars

The strain measurement system is made up of three bars and the test specimen. The incident bar is the bar that the striker comes in contact with when shot. The test specimen is held between the incident bar and the transmitter bar. At the end of the assembly there is a momentum bar that absorbs the remaining energy from the

striker bar. All bars are made of steel and are 18.25 inches long with a diameter of .5 inch.



Fig.9 Incident and transmitter bar

Stand

We decided to use a 10-foot long, 4x4 inch I-beam as a reference plane that most the apparatus sits on. Only gas tank and pressure regulator are not located on the I-beam. The apparatus takes up roughly 3 feet of the beam. The rest of the beam is used to support the setup that the striker bar hits. Five legs made from hollow two by four inch square tubing found in Crosby Lab support the beam.



Fig.10 Stand assembly.

CONTROL SYSTEM

Control Box

The control box houses the electronics needed to trigger and close the solenoid valve. It includes relays, wires, fuse, inlets and outlets for signal receiving processing and delivery to and from computer vacuum pump and Photo-sensor. Data in Appendix 3



Fig.11 Control Box.

Photo Sensor

The Photo sensor utilizes a potentiometer in series with a photo resistor, which in operation varies the voltage across the Photo sensor depending upon the intensity of light on its surface, which can also be tuned using the potentiometer. This voltage is high when light is incident on its surface and low when the sensor is blocked by a projectile. Changes in voltage are processed and

delivered to the computer that shuts off the solenoid valve and cutting off the nitrogen supply.

Laser

We used small commercially available hunting laser with a power level of roughly 5mW and is powered by 3 small 1.5 volts batteries or by an external DC voltage supply. Below are two pictures of the laser and photo sensor that show how the striker comes out and blocks the laser, which changes the voltage read by the control system and turns the solenoid valve off.

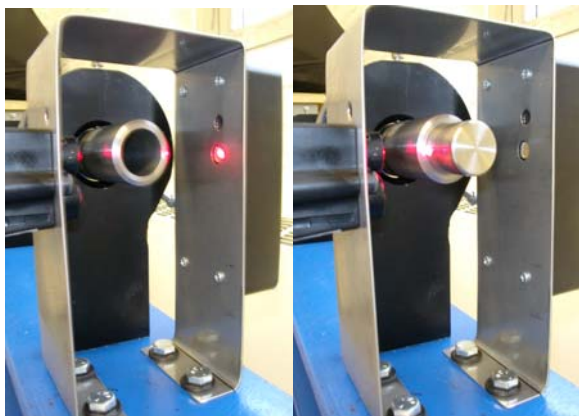


Fig.12 On the left laser active. On the right striker bar exiting the tube and disrupting the laser light

Daqbook

A Daqbook Iotech Daqbook 100, is used as the focal point of our control system. The daqbook is controlled by a Delphi programs which access several features such an Intel 9513 Timer Chip, Digital I/O ports for controlling relays and several analog ports for taking voltage readings.

It ties all other systems together in order to fire the gas gun.



Fig.13 Daqbook

As mentioned earlier Daqbook is connected to the computer and the control box allowing for signal processing



Fig.14 Computer connected to the Daqbook

Control Program

The control program is written Delphi 5 and is derived from software used by Professor Vincent Caccese for other applications. The program uses Windows dlls to control Daqbook functions which can then be accessed by our control program written in Delphi. It contains functions which read voltages and turn on/off relays. Using these features in combination makes it possible to

control the Split Hopkinson Bar Apparatus. Programs are listed in Appendix 4.

Control Program Main Form

The following figure is a picture of the main screen to control Split Hopkinson Bar Apparatus. In the picture you can see where the solenoid valve and vacuum pump are turned on an off, the voltage readout for the laser at the end of the barrel, and the all important button which fires the striker.

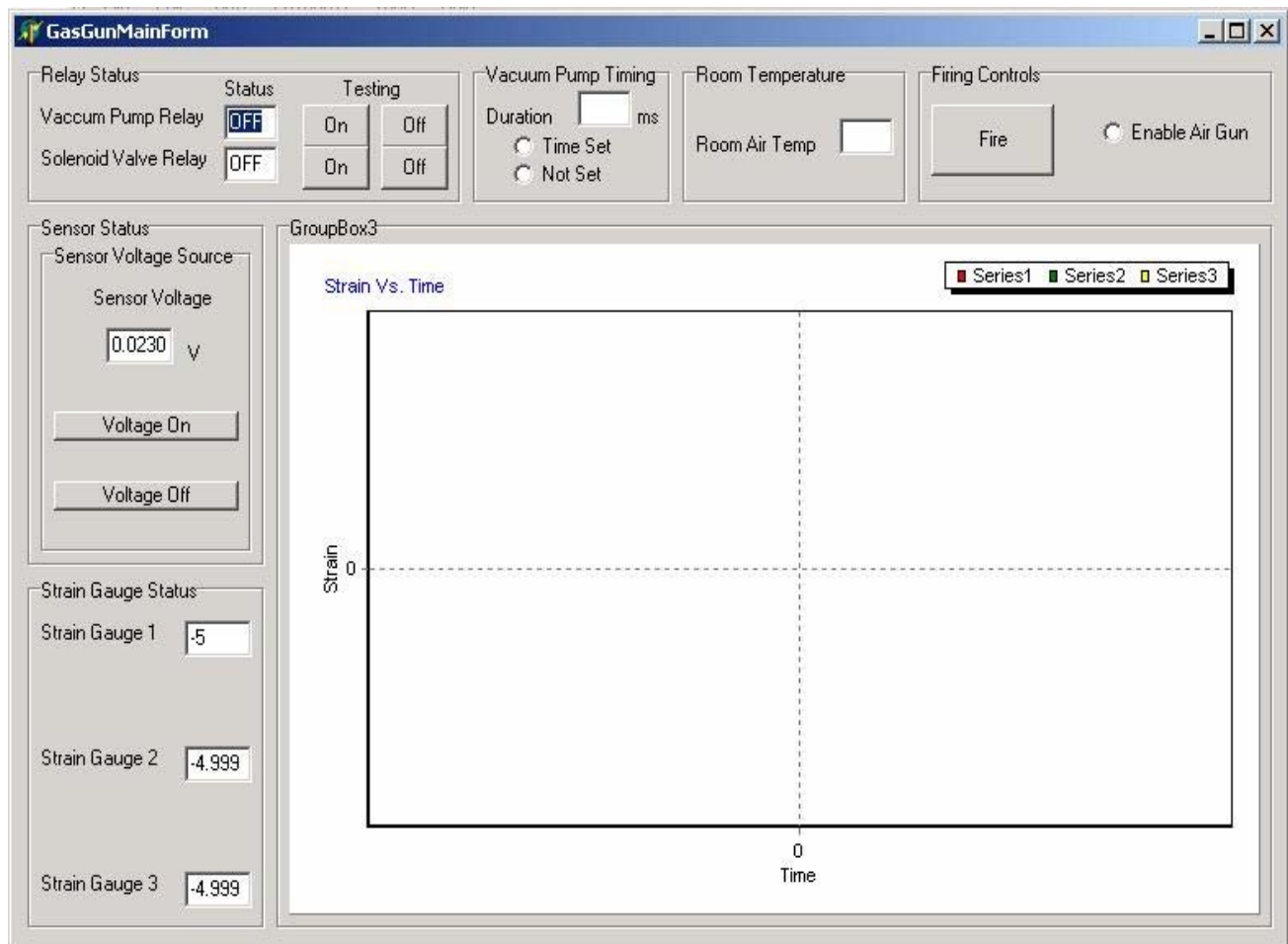


Fig.15 Control Program Main Form

DESIGN TESTING-EXPERIMENTAL

Velocity Calibration

First test performed with the Split Hopkinson Bar Apparatus was velocity calibration. Pressure controller was used to incrementally release a certain amount of gas to the solenoid valve. The test was performed using pressures ranging from 25 psi to 550 psi. The calibration test was done to see what velocities would be reached for corresponding various shooting pressures. We performed this procedure several times to obtain the repeatability of results. The program described in the previous section allowed us to read the results: velocity and travel time off the computer screen thanks to the interface. We obtained three sets of values during our experiments. Later on those values were used to produce calibration curve using Microsoft Excel. Values obtained during those three experiments are listed in the table that follows.

Velocity Calibration Test Results			
Test Date	Pressure (psig)	Travel Time (sec)	Velocity (ft/sec)
04-19-2007	50	0.094	21.2766
	110	0.095	21.05263
	165	0.07	28.57143
	250	0.07	28.57143
	360	0.059	33.89831
	450	0.06	33.33333
	550	0.062	32.25806
04-30-2007	25	0.131	15.26718
	50	0.092	21.73913
	100	0.082	24.39024
	150	0.074	27.02703
	200	0.074	27.02703
	300	0.064	31.25
05-01-2007	75	0.089	22.47191
	85	0.081	24.69136
	90	0.09	22.22222
	140	0.077	25.97403
	150	0.074	27.02703
	180	0.071	28.16901
	220	0.07	28.57143
	260	0.06	33.33333

Table 2. Shows calibration data

Using the values from table above, a velocity versus pressure graph was made that is located in the figure on the following page. On the chart the best fitted line is $y = 7.9507x^{0.2387}$ This line gives the best estimate of what the striker velocity will be for a given pressure.

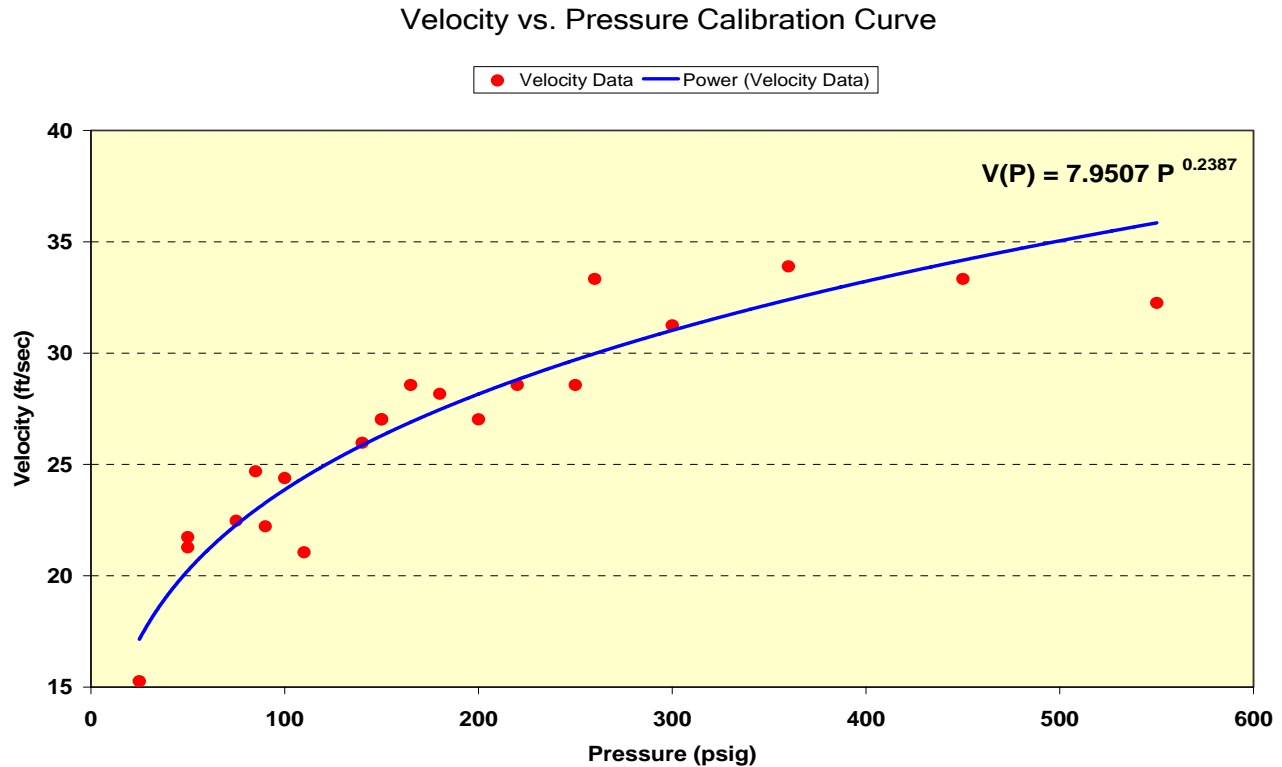


Fig.16 Shows Velocity-Pressure Calibration Curve for the Split Hopkinson Bar Apparatus

Strain Measurements

The strain-stress measurement was performed at the same time as the velocity test. Strain is measured in the incident and first transmitter bars. Three strain gages were strategically located on the bars, two on the incident bar, and one on the transmitter bar. The strain gages measured the wave that propagated through the bars. When the striker hits the incident bar, the wave travels through the bar until it reaches the test specimen. Depending on the specimen, some of the wave reflects back through the incident bar while the

rest travels through the specimen and through the transmitter bar. For the strain test, the strain gages were connected to a P-3500 strain amplifier and the output was shown on a Hewlett Packard Oscilloscope.

The following two figures⁴ located on the next page show the expected waves traveling through the incident and transmitter bars. The figures following are the images of propagation waves gathered during our experiment. They are given as copy of the printout and also pictures taken from the oscilloscope screen.

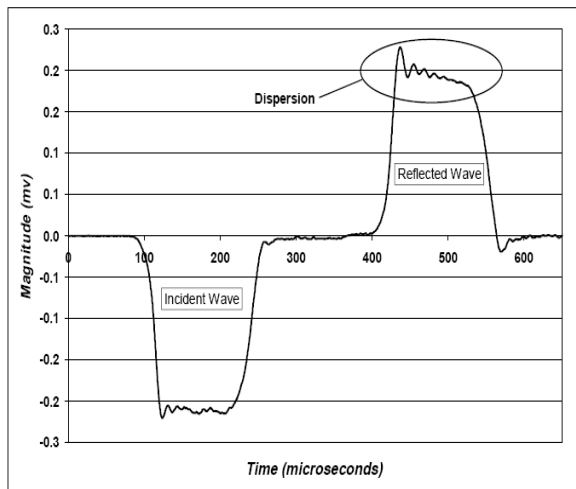


Fig .17: Expected Incident Bar Wave

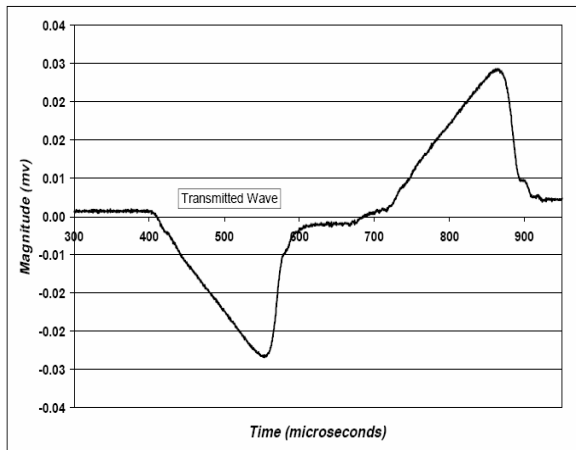


Fig.18: Expected Transmitter Bar Wave

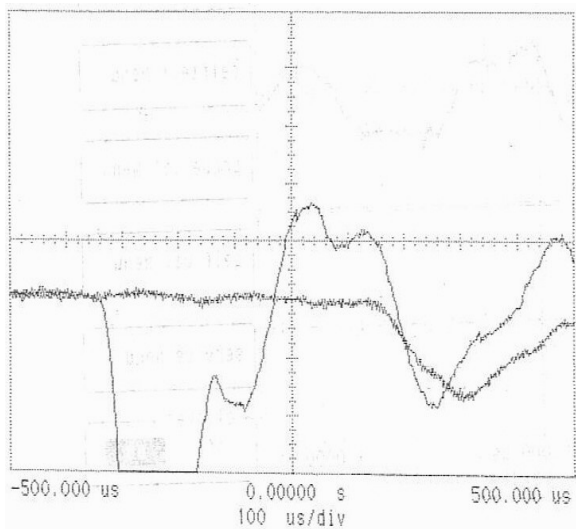


Fig 19: Wave in Incident and Transmitter Bars with Composite Test Specimen

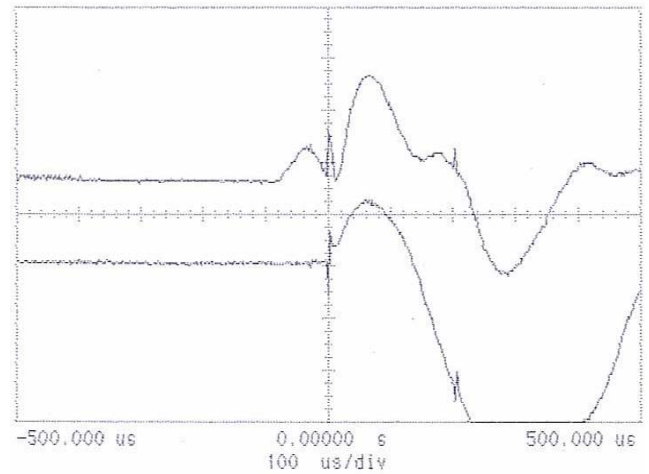


Fig.20 Strain Wave in Incident and Transmitter Bars with Composite Test Specimen shot at 260psi

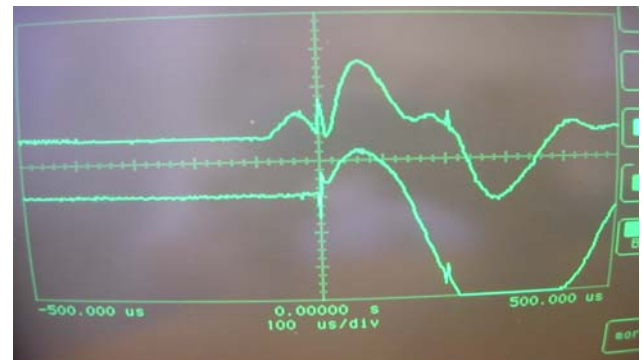


Fig 21: Wave in Incident and Transmitter Bars with Composite Test Specimen shot at 260psi

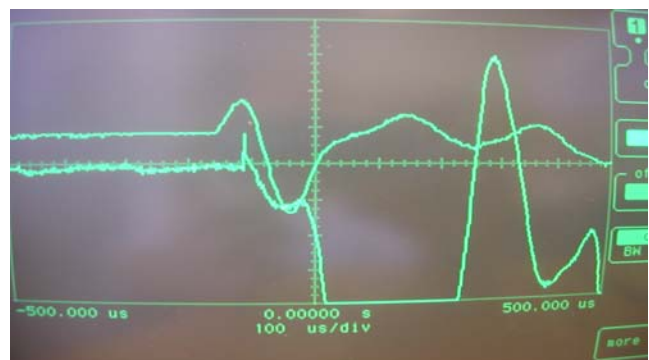


Fig 22: Wave in Incident and Transmitter Bars with Composite Test Specimen shot at 180psi

After problems arose with the strain gages on the incident bar, only the wave from the strain gage on the transmitter bar could be seen. The rest of the figures are listed in Appendix 2.

Results and Conclusion

Overall the designing and building of the Split Hopkinson Pressure Bar Apparatus was a success. We went from the thought of just designing the gun at the beginning of the year to full building and testing at the end of last month. Although we were not able to obtain the predicted velocities we reached striker velocities in the range of over 35 ft/sec while not even shooting at full potential. During the same velocity test we were able to gather several strain curves from an oscilloscope, which compared fairly well to those found in the military booklet and others online. Possible reasons for reduced velocities:

1. Design calculations were based on the flow in a straight pipe - not through converging nozzle, but reduced throat area was necessary to keep the striker bar in the right position.
2. Design calculations did not include air frictional factors-back pressure.
3. Friction of the striker bar against the barrel was not included in estimation.
4. Experimental velocities were reduced by the uncertainty connected to computer hardware. Because a minimum time increment of the counter was 15 milliseconds at high pressures did not give enough room to measure

elapsed time. Evidently, this design class and project have given us the best learning experience. Undoubtedly we will encounter similarly challenging tasks in the real engineering world

Recommendations

1. To allow for the diversification of testing out technical advisor Professor Vince Caccese suggested putting a steel plate at the end of the barrel. It will allow holding an object that would undergo destructive testing.
2. More velocity calibration test performed at larger pressures is required. At the same time find a way to get the time increments smaller in the computer program for better velocity accuracy.
3. Search for better strain gages. Testing at high pressures and velocities broke the gages located on the incident bar; therefore more durable ones should be used in the future.
4. It will absolutely essential to resolve the counter problem, so the system can use smaller time increments to do the measurements.

APPENDIX 1

The following are the transitional ideas-options for the various elements of the apparatus that were put aside during the process and reasons for these decisions.

Piston

Originally the group considered the idea of using a piston with a striker bar part connected to it to allow the piston much higher accelerations and a much smaller barrel length. This idea was scrapped in favor of a longer barrel.

Fabricated Pressure Vessel

Early in the project it was decided that the air supply for the air gun would be a pressure vessel made of a hollow tube and side plates. Initial dimensions of 8.5 inches in diameter and 11 inches long were chosen for initial performance calculations, but the idea was scrapped in favor of prefabricated compressed gas tanks for safety and ease of use.

Maximum Optimal Sized Barrel

The original plan for the group was to use the general dimensions of most other split hopkinson bar projects and build something

around the same dimensions we found in other sources. Our dimensions were later scaled down due to inability to produce or subcontract the manufacturing of a very long barrel with a very tight tolerances and within a reasonable budget limits. At the beginning of the semester, we looked into buying a barrel that met the requirements of the military specifications. The size of the barrel was 100 inches long with a 1-inch inner diameter and 3 inch outer diameter. The 1-inch thickness was to meet the straightness requirement of the barrel and not a safety factor. After talking several machine shops that could not do the drilling until late 2007, or who gave quotes exceeding our budget we decided to downsize. The barrel we have chosen was an original part of a shoot gun drilled gun barrel, which was 30 inches long, has an inner diameter of 0.71 inches. Although the cost of this barrel was very low comparing to the previous options, final on site inspection of the tube excluded it from a potential application. It turned out that the inside diameter was tapered (0.712 to 0.725), therefore making it unable to have the desired tolerances.

Mylar Discs

Initially we were considering using mylar membranes/disks that are melted by running

a current through a small wire in contact with it to release the pressure stored in their air tank. The idea was abandoned due to the possibility of clogging of the launch tube by the broken disks.

in. It may however be an excellent idea for next semester were time permitting.

Iris Valves

Considering affordable iris valves we discovered that iris valves are made to handle gravity fed systems and would not have stood up to the high pressures that it would have been subjected to in our apparatus.

Rupture Disks

Another option for a fast release of gas from the reservoir to the launch tube was an aluminium rupture disk. Rupture disks would have been an excellent method seeing as there is practically no opening time as with solenoid valves, but the cost per disk and high uncertainty with burst pressure eliminated it.

Backup Gas Release

It was suggested midway through the project that our group should have a backup system for gas release in case of a solenoid valve failure. This wasn't entirely a dead end or a scrapped idea per say, but it just wasn't something we could afford to invest the time

APPENDIX 2

More Strain Measurement Graphs

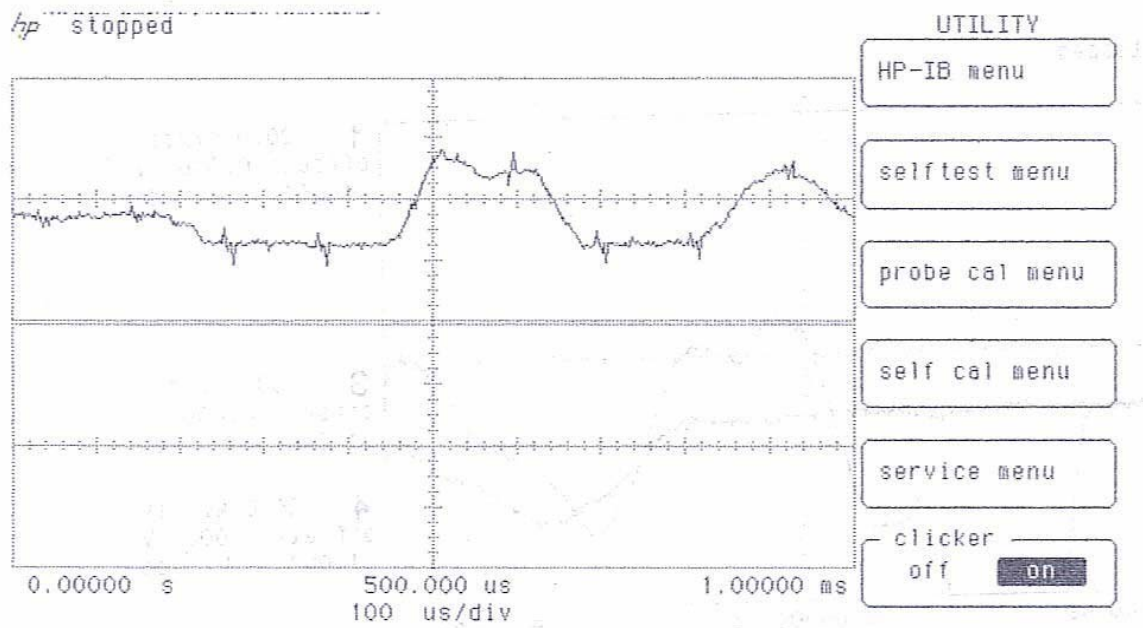


Fig 23: Wave in Incident Bar with Composite Test Specimen

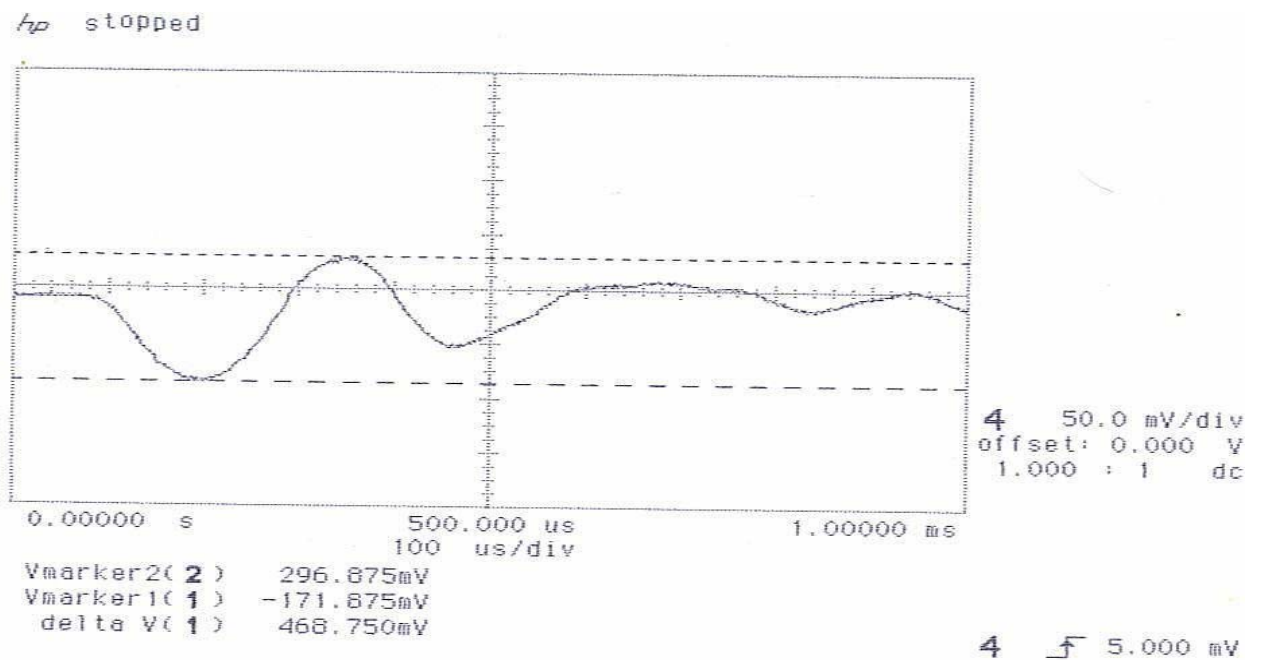


Fig 24: Wave Through Transmitter Bar shot at 260 psi

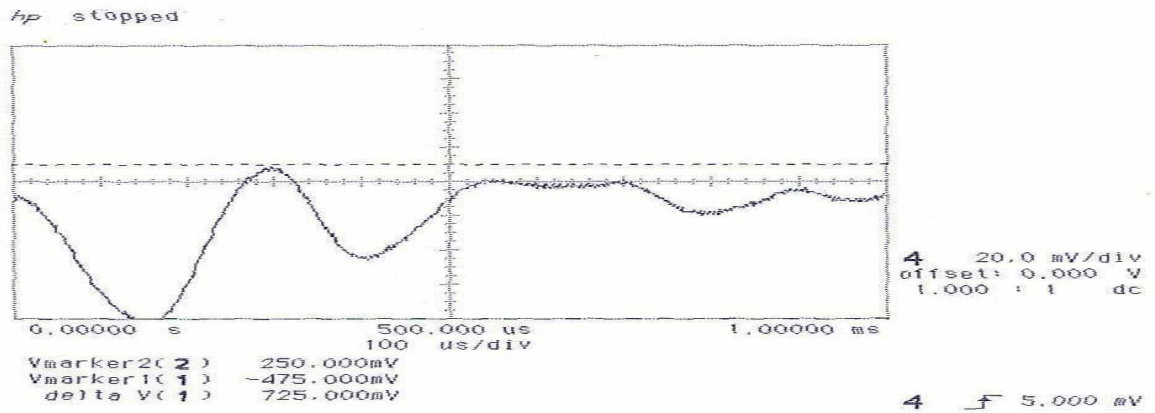


Fig 25: Wave Through Transmitter Bar shot at 220 psi

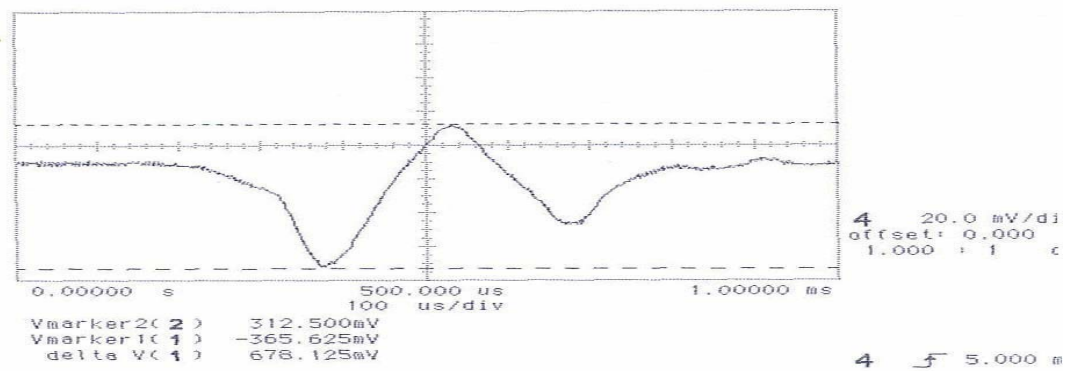


Fig 26: Wave Through Transmitter Bar shot at 180 psi

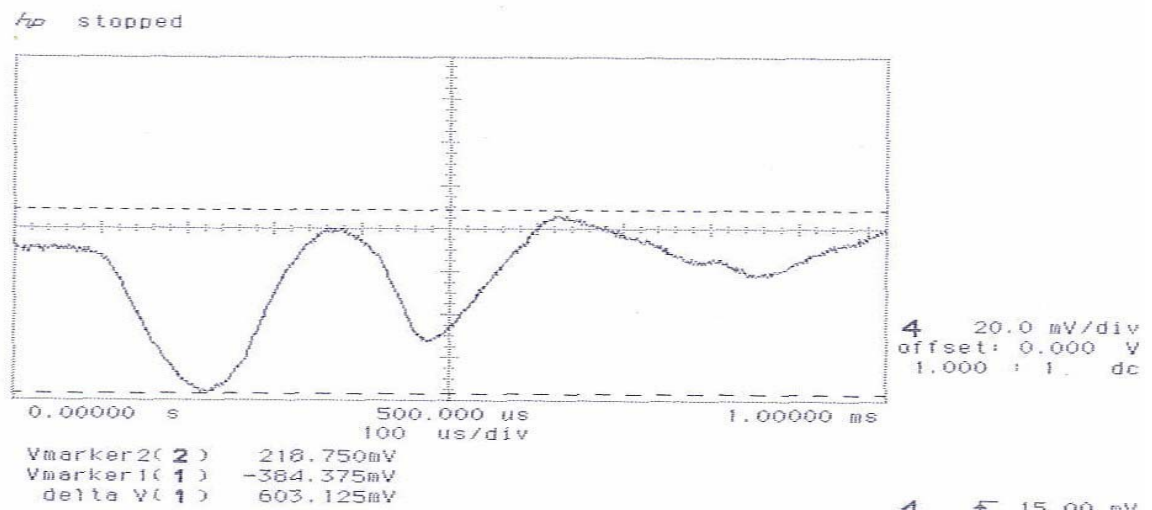


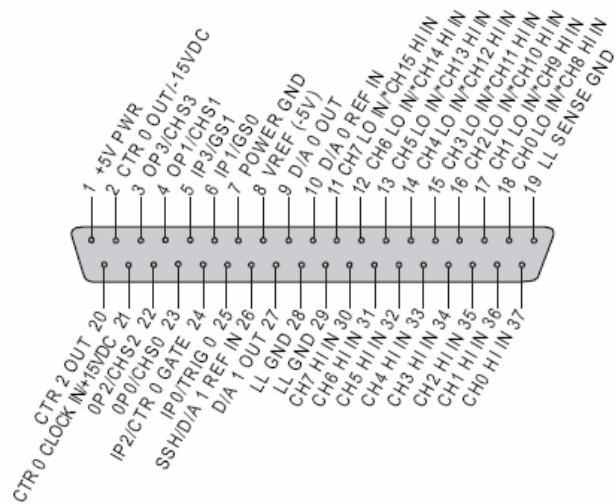
Fig 27: Wave Through Transmitter Bar shot at 150 psi

APPENDIX 3

Shows connections of the Analog and Digital Ports in the control Box. Please refer to the DAQBOOK 00 Users manual for further reference.

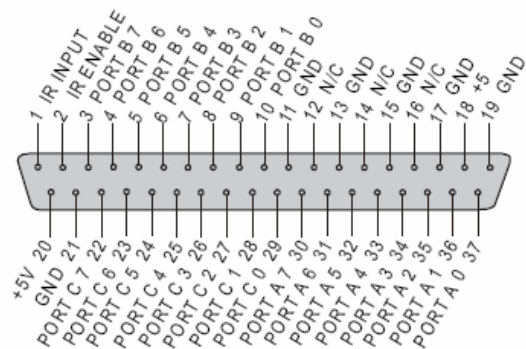
1. ANALOG PORT (front View)

DaqBook
P1 Pinout
Analog I/O
(compatible with
Metrabyte DAS-16)



2. DIGITAL PORT (front View)

DaqBook
P2 Pinout
Digital I/O
This P2 interface is
available on the
DaqBook/100, /120,
/200, and /260.



3. OUR CONNECTIONS (Back View-from the iside of the Control Box)

PIN CONNECTOR NUMBER	ANALOG I/O	DIGITAL I/O
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		fan voltage (digital ground)
12		
13		
14		
15		Voltage to sensor BLACK,dg
16		
17		
18		+ 5 V supply to fan
19	Signal from sensor-RED	Vacum relay digital ground
20		
21		Solenoid relay digital ground
22		
23		
24		
25		
26		
27		
28	+ from thermocouple	
29	Ground for Strain Gage Input	
30	Signal from sensor-GREEN	
31	- from thermocouple	
32		
33	Strain Gage input from amp	
34	Strain Gage input from amp	
35	Strain Gage input from amp	Digital out to sensor-WHITE
36	Strain Gage input from amp	Dig. Out to Solenoid Relay
37	Strain Gage input from amp	Dig. Out to Vacum Pump Rel

APPENDIX 4

Includes the computer code used to to control Split Hopkinson Pressure Bar Apparatus. The **GasGunC** file defines constants, the **GasGunMain** file initializes the hardware and defines subroutines, the **GasGunInterface** file is the code which the user controls the daq with, the **daqx** file contains the commands that Delphi uses to control the daqbook, and the **Errex** file is used to give error commands from the daqbook to the user. The picture is of the form associated with the interface file that the user interacts with to run subroutines within the interface file.

ENDNOTES

¹ Design and Operating Characteristics of a Split Hopkinson Pressure Bar Apparatus. Army Materials and Mechanics Research. Center Watertown Massachusetts, Report by Kenneth D. Robertson, Shun-Chin Chou and James H. Rainey. November 1971

² www.asco.com

³ www.asco.com

⁴ Advancements in the Split Hopkinson Bar Test, Kaiser, Michael Adam, pg 61-62, <http://scholar.lib.vt.edu/theses/available/etd-41998-18465/unrestricted/ETD.pdf>

```
1: unit GasGunC;
2:
3: interface
4: uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
5:       Dialogs ,DaqX;
6:
7: const
8: DIOPort_A      = 0;
9: DIOPort_B      = 1;
10: DIOPort_C      = 2;
11:
12: DIO_Write      = false;
13: DIO_Read       = true;
14:
15: VacuumPumpRelay = 0;
16: SolenoidValveRelay = 1;
17: SensorVoltageSource = 2;
18: SensorChan = 7;
19: Gauge1Chan = 0;
20: Gauge2Chan = 1;
21: Gauge3Chan = 2;
22:
23:
24: var
25:   DaqHandle00      : DaqHandleT;
26:   daqDeviceType00  : DaqDacDeviceType;
27:   daqIODeviceType00 : DaqIODeviceType;
28:   DIOConfig        : byte;
29:   DIODevPort       : DWORD;
30:   DIODevType       : DaqIODeviceType;
31:   DIOExpansionPort : DaqIOExpansionPort;
32:   DIOWhichDevice   : DWord;
33:
34: implementation
35:
36: end.
```



```
1: unit GasGunMain;
2:
3: interface
4:
5: uses
6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
7:   StdCtrls, ExtCtrls, TeeProcs, TeEngine, Chart, GasGunInterface,
8:   Errex, GasGunC, AppEvnts, daqx, Series;
9:
10: type
11:   TGasGunMainForm = class(TForm)
12:     RelatStatuGB: TGroupBox;
13:     vrs: TEdit;
14:     Label1: TLabel;
15:     Label2: TLabel;
16:     srs: TEdit;
17:     SensorStatusGB: TGroupBox;
18:     StrainGaugeStatusGB: TGroupBox;
19:     Label4: TLabel;
20:     Label6: TLabel;
21:     Label7: TLabel;
22:     StrainGauge1VoltageEB: TEdit;
23:     StrainGauge2VoltageEB: TEdit;
24:     StrainGauge3VoltageEB: TEdit;
25:     RoomtempGB: TGroupBox;
26:     RoomAirTempEB: TEdit;
27:     Label8: TLabel;
28:     GroupBox1: TGroupBox;
29:     FireControlBT: TButton;
30:     SensorVoltageSourceGB: TGroupBox;
31:     Label9: TLabel;
32:     SensorVoltageEB: TEdit;
33:     VoltageOnBT: TButton;
34:     VoltageOffBT: TButton;
35:     VacPumpOn: TButton;
36:     VacPumpOff: TButton;
37:     SolValveOff: TButton;
38:     SolValveOn: TButton;
39:     Label10: TLabel;
40:     Label11: TLabel;
41:     RadioButton1: TRadioButton;
42:     Label14: TLabel;
43:     GroupBox2: TGroupBox;
44:     Edit1: TEdit;
45:     Label16: TLabel;
46:     RadioButton3: TRadioButton;
47:     RadioButton4: TRadioButton;
48:     Label15: TLabel;
49:     Timer2: TTimer;
50:     GroupBox3: TGroupBox;
51:     Chart1: TChart;
52:     Series1: TPointSeries;
53:     Series2: TPointSeries;
54:     Series3: TPointSeries;
55:     procedure FormCreate(Sender: TObject);
56:     procedure VacPumpOnClick(Sender: TObject);
57:     procedure VacPumpOffClick(Sender: TObject);
58:     procedure SolValveOnClick(Sender: TObject);
59:     procedure SolValveOffClick(Sender: TObject);
60:     procedure VoltageOnBTClick(Sender: TObject);
61:     procedure VoltageOffBTClick(Sender: TObject);
62:     procedure Timer2Timer(Sender: TObject);
63:     procedure FireControlBTClick(Sender: TObject);
64:   private
65:     { Private declarations }
66:   public
67:     { Public declarations }
68:   end;
69:
70: var
71:   GasGunMainForm: TGasGunMainForm;
```

```
72:   Fired:           Integer;
73:   Counter:         Integer;
74:   SensorVoltage:   Real;
75:   StrainGauge1Voltage: Real;
76:   StrainGauge2Voltage: Real;
77:   StrainGauge3Voltage: Real;
78:
79: implementation
80:
81: {$R *.DFM}
82:
83: procedure TGasGunMainForm.FormCreate(Sender: TObject);
84: begin
85:   OpenTheDaq(1);
86:
87:   daq9513SetMasterMode(DaqHandle00, DiodeLocal9513, 0, 1, DcsF4, false, false,
88:     DtodDisabled);
89:
90:   daq9513SetCtrMode (DaqHandle00, DiodeLocal9513, 0, 1, DgcNoGating, true,
91:     DcsF4, false, false, false, false, true, DocInactiveLow);
92:
93:   GasGunInterface_Frm.InitRelays;
94:   {Turning all relay status settings to off}
95:   vrs.text := 'OFF';
96:   srs.text := 'OFF';
97:   Counter := 0;
98:
99: end;
100:
101: {Turning on Vacuum Pump Relay via button}
102: procedure TGasGunMainForm.VacPumpOnClick(Sender: TObject);
103: begin
104:   GasGunInterface_Frm.TurnRelayOn(VacuumPumpRelay);
105:   vrs.text := 'ON';
106: end;
107:
108: {Turning off Vacuum Pump Relay via button}
109: procedure TGasGunMainForm.VacPumpOffClick(Sender: TObject);
110: begin
111:   GasGunInterface_Frm.TurnRelayOff(VacuumPumpRelay);
112:   vrs.text := 'OFF';
113: end;
114:
115: {Turning on Solenoid Valve Relay via button}
116: procedure TGasGunMainForm.SolValveOnClick(Sender: TObject);
117: begin
118:   GasGunInterface_Frm.TurnRelayOn(SolenoidValveRelay);
119:   srs.text := 'ON';
120: end;
121:
122: {Turning off Solenoid Valve Relay via button}
123: procedure TGasGunMainForm.SolValveOffClick(Sender: TObject);
124: begin
125:   GasGunInterface_Frm.TurnRelayOff(SolenoidValveRelay);
126:   srs.text := 'OFF';
127: end;
128:
129: procedure TGasGunMainForm.VoltageOnBTClick(Sender: TObject);
130: begin
131:   GasGunInterface_Frm.TurnRelayOn(SensorVoltageSource);
132: end;
133:
134: procedure TGasGunMainForm.VoltageOffBTClick(Sender: TObject);
135: begin
136:   GasGunInterface_Frm.TurnRelayOff(SensorVoltageSource);
137: end;
138: {*****}
139: *      Firing Controls      *
140: {*****}
141:
142: procedure TGasGunMainForm.Timer2Timer(Sender: TObject);
```

```
143: begin
144:   SensorVoltage := Scandata.ReadOneChannel(SensorChan);
145:   SensorVoltageEB.Text := FloatToStr(SensorVoltage);
146:   StrainGauge1Voltage := Scandata.ReadOneChannel(Gauge1Chan);
147:   StrainGauge1VoltageEB.Text := FloatToStr(StrainGauge1Voltage);
148:   StrainGauge2Voltage := Scandata.ReadOneChannel(Gauge2Chan);
149:   StrainGauge2VoltageEB.Text := FloatToStr(StrainGauge2Voltage);
150:   StrainGauge3Voltage := Scandata.ReadOneChannel(Gauge3Chan);
151:   StrainGauge3VoltageEB.Text := FloatToStr(StrainGauge3Voltage);
152: end;
153:
154:
155: procedure TGasGunMainForm.FireControlBTCClick(Sender: TObject);
156: var
157:
158:   loop : boolean;
159:   SensorV : real;
160:   str1 : string;
161:   str2 : string;
162:   str3 : string;
163:   DateTime : TDateTime;
164:   Hour, Min1, Min2, Sec1, MSec1, Sec2, MSec2: Word;
165:   eTime : real;
166:   counts : WORD;
167:   count1 : WORD;
168:   count2 : WORD;
169:   count3 : WORD;
170:   something : WORD;
171:   velocity: real;
172:   inc : word;
173:
174: begin
175:   If Radiobutton1.Checked Then
176:   Begin
177:     loop := true;
178:     count1 := 0;
179:     count2 := 0;
180:     count3 := 0;
181:
182:     series1.Clear;
183:     series2.Clear;
184:     series3.Clear;
185:
186:     //DecodeTime(Time,Hour,Min1,Sec1,MSec1);           //System Based Time Measurement
187:
188:     GasGunInterface_Frm.TurnRelayOn(SolenoidValveRelay);
189:
190:     daq9513SetHold(DaqHandle00, DiodtLocal9513,0, 1, 0);
191:
192:     daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccArm, true, false, false,
193:     false, false);
194:
195:     daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccSave, true, false, false,
196:     false, false);
197:
198:     daq9513GetHold(DaqHandle00, DiodtLocal9513, 0, 1, count1);
199:
200:     while loop
201:     do
202:     begin
203:       daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccSave, true, false, false,
204:       false, false);
205:       daq9513GetHold(DaqHandle00, DiodtLocal9513, 0, 1, count2);
206:       SensorV := Scandata.ReadOneChannel(SensorChan);
207:       if SensorV > 3.5 then Loop := false;
208:     end;
209:
210:     daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccDisarm, true, false, false,
211:     false, false);
212:
213:     GasGunInterface_Frm.TurnRelayOff(SolenoidValveRelay);
```

```
214:
215: {DecodeTime(Time,Hour,Min2,Sec2,MSec2);           //System Time Based measurement
216:
217: eTime := (Sec2-Sec1) + (MSec2-Msec1)/1000;} //System Time Based measurement
218:
219: If (count2-count1) > 0 Then
220: begin
221: str1 := 'Projectile Velocity ' + FloatToStr(2/((count2 - count1)/1000)) + ' ft/s';
222: str2 := 'Projectile Travel Time ' + FloatToStr((count2-count1)/1000) + 's';
223: showmessage(str1);
224: showmessage(str2);
225: end
226: else
227: If (count2-count1)=0 Then
228: Begin
229: str1 := 'Projectile Velocity - ' + 'infinity' + ' ft/s';
230: str2 := 'Projectile Travel Time ' + '0' + 's';
231: showmessage(str1);
232: showmessage(str2);
233:
234: radiobutton1.Checked := False;
235:
236: End;
237: {*****
238: *                               Strain Gauge Measurements                               *
239: *****}
240:
241: loop := true;
242: something :=1;
243:
244: daq9513SetHold(DaqHandle00, DiodtLocal9513,0, 1, 0);
245:
246: daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccArm, true, false, false,
247: false, false);
248:
249: daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccSave, true, false, false,
250: false, false);
251:
252: daq9513GetHold(DaqHandle00, DiodtLocal9513, 0, 1, count1);
253:
254: while loop
255: do
256: begin
257:   daq9513MultCtrl(DaqHandle00, DiodtLocal9513, 0, DmccSave, true, false, false,
258:   false, false);
259:   daq9513GetHold(DaqHandle00, DiodtLocal9513, 0, 1, count2);
260:   StrainGauge1Voltage := Scandata.ReadOneChannel(Gauge1Chan);
261:   Series1.AddXY( (count2-count1), StrainGauge1Voltage);
262:   Series2.AddXY( (count2-count1), StrainGauge2Voltage);
263:   Series3.AddXY( (count2-count1), StrainGauge3Voltage);
264:   if count2 - count1>100 then Loop := false;
265: end;
266:
267: end
268:
269: end;
270:
271: end.
```

```
1: unit GasGunInterface;
2:
3: interface
4:
5: uses
6:   Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
7:   Dialogs, Daqx, GasGunC;
8:
9: Const
10:     MaxChannels    = 16;
11:     ADVOLTS        = 5;
12:
13: var
14:   DaqPort          : word;
15:   err               : DaqError;
16:
17: procedure CreateScanDataObject(StartChan,EndChan:DWORD; pts:longint;
18:                               Freq:Single; Ga : byte);
19: type
20:   PScanData = ^TScanData;
21:   TScanData = array [0..1] of Word;
22:
23:   PScanTags = ^TScanTags;
24:   TScanTags = array [0..1] of Byte;
25:
26:   PRealData = ^TRealData;
27:   TRealData = array [1..MaxChannels,0..1] of Single;
28:
29: type
30:   PReadScanObject = ^TReadScanObject;
31:   TReadScanObject = object
32:     NoOfChannels      : Integer;
33:     StartChannel,
34:     EndChannel        : DWORD;
35:     PointsToAcquire   : longint;
36:     Frequency          : single;
37:     Gain               : byte;
38:     OneShot            : byte;
39:     Trigger            : word;
40:     Level              : byte;
41:     CalFactor          : array [1..MaxChannels] of Single;
42:     constructor Init(StartChan,EndChan:DWORD; pts:longint;
43:                     Freq:Single; Ga : byte);
44:     destructor Done;
45:     function  ReadOneChannel(ChNo: integer): real;
46:
47:   protected
48:     TotalData          : longint;
49: end;
50:
51: var
52:   ScanData          : PReadScanObject;
53:   RawData            : PScanData;
54:   Buffer              : PScanData;
55:   Tags               : PScanTags;
56:   RealData           : PRealData;
57:
58: type
59:   TGasGunInterface_Frm = class(TForm)
60:   private
61:     { Private declarations }
62:   public
63:     { Public declarations }
64:     procedure InitRelays;
65:     procedure TurnRelayOn(ChNo : integer);
66:     procedure TurnRelayOff(ChNo : integer);
67:   end;
68:
69: var
70:   GasGunInterface_Frm: TGasGunInterface_Frm;
71:
```

```
72: implementation
73:
74: {$R *.DFM}
75: {*****}
76: procedure TGasGunInterface_Frm.InitRelays;
77: var
78:     cfg : DWord;
79:     err : DaqError;
80: begin
81:     if daqIOGet8255Conf(DaqHandle00,DIO_Write,DIO_Write,DIO_Read,DIO_Write,cfg) <>
82:         DerrNoError then showmessage('DIO Port ERROR');
83:         DIOConfig      := cfg;
84:         DIODevType      := DiodtLocal8255;
85:         DIODevPort      := Diodp8255A;
86:         DIOExpansionPort := Dioep1;
87:         DIOWhichDevice   := 0;
88:
89: end;
90: {*****Turning Relays On*****}
91: procedure TGasGunInterface_Frm.TurnRelayOn(ChNo : integer);
92: var
93:     error      : DaqError;
94:     bitValue    : longbool;
95:     cfg : Dword;
96: begin
97:         bitValue :=true; {Ord(true)=1}
98:         error := daqIOWriteBit(DaqHandle00,DIODevType,
99:             DIODevPort,DIOWhichDevice,DIOExpansionPort,
100:             ChNo,bitValue);
101: end;
102: {*****Turning Relays Off*****}
103: procedure TGasGunInterface_Frm.TurnRelayOff(ChNo : integer);
104: var
105:     error      : DaqError;
106:     bitValue    : longbool;
107:     cfg : Dword;
108: begin
109:         bitValue := false; {Ord(false)=0}
110:         error := daqIOWriteBit(DaqHandle00,DIODevType,
111:             DIODevPort,DIOWhichDevice,DIOExpansionPort,
112:             ChNo,bitValue);
113: end;
114:
115: procedure CreateScanDataObject(StartChan,EndChan:DWORD; pts:longint;
116:     Freq:Single; Ga : byte);
117:
118: begin
119:     New(ScanData, Init(StartChan,EndChan,pts,Freq,Ga));
120: end;
121:
122: constructor TReadScanObject.Init(StartChan,EndChan:DWORD; pts:longint;
123:     Freq:Single; Ga : byte);
124:
125: begin
126:
127:     StartChannel      := StartChan;
128:     EndChannel        := EndChan;
129:     NoOfChannels      := EndChannel-StartChannel+1;
130:     PointsToAcquire   := pts;
131:     Frequency         := Freq;
132:     Gain              := Ga;
133:     OneShot           := 0;
134:     Level             := 0;
135:     Trigger           := 0;
136:     TotalData         := NoOfChannels*PointsToAcquire;
137:     GetMem(RawData,   TotalData*SizeOf(Word));
138:     GetMem(Buffer,    TotalData*SizeOf(Word));
139:     GetMem(Tags,      TotalData*SizeOf(Byte));
140:     GetMem(RealData,  TotalData*SizeOf(Single)*MaxChannels);
141: end;
142:
```

```
143: destructor TReadScanObject.Done;
144: begin
145: end;
146:
147: function TReadScanObject.ReadOneChannel(ChNo: integer): real;
148: var
149:   Flags   : integer;
150:   Gains    : integer;
151:   sample   : word;
152:   Temp1    : real;
153: begin
154:   Gains := DgainX1;
155:   Flags := DafAnalog + DafSingleEnded + DafBipolar;
156:   daqAdcRd(DaqHandle00,ChNo,sample,Gains,flags);
157:   { sample := sample shr 0; }
158:   Temp1 := (sample-2048*16)*(ADVOLTS)/(2048*16);
159:   Result := Temp1;
160: end;
161:
162: end.
```

```
1: unit Errex;
2:
3: interface
4:
5: uses DaqX, GasGunInterface, GasGunC;
6:
7:
8:
9: const
10: DV_DaqBook00 = 0;
11:
12:
13: procedure TestError(Device: integer);
14: procedure SetTheErrorHandler;
15: procedure OpenTheDaq(Device: integer);
16:
17: // Daqx error handler prototypes
18: procedure ErrorHandler( errCode: DaqError ); stdcall;
19:
20: implementation
21:
22: uses SysUtils, Dialogs;
23:
24:
25: *****
26: // This unit demonstrates how to initialize a daqx device and
27: // how to select and error handler
28: //
29: // Functions used:
30: // daqOpen( daqName )
31: // daqOnline( handle, online )
32: // daqSetDefaultErrorHandler( errHandler )
33: // daqSetErrorHandler( handle, errHandler )
34: // daqAdcSetTrig( handle, triggerSource, rising, level, hysteresis, channel );
35: // daqClose( handle )
36: //
37: procedure TestError(Device: integer);
38: var
39:     online: longbool;
40:     { drVersion: DWORD;           // Use to receive driver version }
41:     { hwVersion: DWORD;          // Use to receive hardware version }
42: begin
43:
44: end;
45:
46: *****
47: // Error handler for daq errors.
48: //
49: procedure ErrorHandler( errCode: DaqError ); stdcall;
50: var
51:     msg: string;
52: begin
53:     if ord(errCode)=39 then
54:     begin
55:         ShowMessage('ERROR - Thermocouple Malfunction');
56:
57:     end
58:     else
59:     begin
60:         msg:= 'Message from procedure ErrorHandler' + Chr(13) + Chr(10);
61:         msg:= msg + 'DaqComp error number ' + IntToStr(ord(errCode)) + ' occurred.';
62:         MessageDlg(msg , mtError, [mbOk], 0);
63:     end;
64:
65: end;
66:
67:
68: *****
69: // Procedure to set the error handler. Complains and stops if unable.
70: //
71: procedure SetTheErrorHandler;
```



```
72: begin
73:   (* if (daqSetErrorHandler(addr(ErrorHandler)) <> DerrNoError ) then
74:     begin
75:       MessageDlg('Unable to set default error handler. Exiting program', mtError, [mbOk], 0);
76:       Halt;
77:     end;
78:   HeatChamberMainFrm.Memo1.Lines.Add( 'Error handler set to procedure "ErrorHandler"');
79:   // Disable the choice for the driver built-in error handler since it isn't available
80:   // after setting another handler until the DAQX.DLL is reloaded.
81:   HeatChamberMainFrm.DefaultErrBtn.Enabled:= False;
82:
83: *)
84: end;
85:
86: //*****
87: // Procedure to open a DaqBook or DaqBoard depending on which is selected on Form1.
88: //
89: procedure OpenTheDaq(Device: integer);
90: var
91:   err:      DaqError;
92: begin
93:   DaqHandle00 := daqOpen('DaqBook0');
94: end;
95:
96:
97: end.
```